Design of an RTLinux Motor Control System for Forced Wrist Rotation Studies with fMRI

MATS UPPSTRÖM

KTH Numerical Analysis
and Computer Science

Master's Degree Project
Stockholm, Sweden 2004

TRITA-NA-E04165

Numerisk analys och datalogi
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Design of an RTLinux Motor Control System for Forced Wrist Rotation Studies with fMRI

MATS UPPSTRÖM

TRITA-NA-E04165

Master's Thesis in Computer Science (20 credits)
at the School of Computer Science and Engineering,
Royal Institute of Technology year 2004
Supervisor at Nada was Örjan Ekeberg
Examiner was Anders Lansner

Design of an RTLinux Motor Control System for Forced Wrist Rotation Studies with fMRI

Abstract

This Master's thesis describes the development of a computer controlled mechatronic system used for medical research with fMRI. The computer forcibly moves the patient's hand via a servo motor and measures the force with which the patient resists this move. The purpose of this research is to study the brain patterns of patients with certain types of brain disorders.

The environment that the system was intended for placed serious restrictions on the design. The sensitivity of the magnetic field of the fMRI unit made it impossible to use other that non-magnetic materials. Electrical equipment was used outside the fMRI room, and forces and data was conveyed through non-interfering materials, such as nylon wires and optic fibres.

A safety analysis was made to ensure that the use of this system will not be harmful to the patients.

The computer uses a real-time operating system to grant real-time capacity to events that require a fast response like acquiring data and acting upon it immediately. A freeware RTOS, RTLinux, was chosen, after a brief survey of the market for RTOS was made.

The system was tested successfully and will be used in research during the fall of 2004.

Design av ett RTLinux-styrt motorsystem för studier av påtvingad handledsrotation med fMRI

Sammanfattning

Denna exjobbsrapport beskriver utvecklingen av ett datorstyrt mekatroniskt system för medicinsk forskning med magnetröntgen. Datorn böjer på patientens handled med hjälp av en servomotor och mäter det motstånd som patienten gör mot rörelsen. Syftet med forskningen är att undersöka hjärnaktiviteten hos patienter med vissa typer av hjärnskador.

Den miljö där systemet skall användas lade hårda restriktioner på designen av systemet. Känsligheten i magnetfältet hos MR-kameran gjorde det omöjligt att använda annat än icke-magnetiska material. All elektrisk utrustning placeras utanför MR-rummet, och krafter och data överförs via nylonlinor och optiska fibrer.

En säkerhetsanalys gjordes också för att säkerställa att systemet inte kan skada patienterna.

Datorn kör ett realtids-operativsystem som ger realtidskapacitet åt de händelser som kräver snabba svar, som att mäta indata och reagera på datat omedelbart. Ett 'freeware' RTOS, RTLinux, valdes efter att ha gjort en översikt över RTOS marknaden.

Systemet testades framgångsrikt och kommer användas i studier under hösten 2004.

Foreword

This Master's thesis was done at the research group for Studies of Artificial Neural Systems (SANS) at the department of Numerical Analysis and Computer Science (NADA) at the Royal Institute of Technology (KTH) in Stockholm, Sweden, and the employer was the department of Neuropediatrics at Astrid Lindgren's Children's Hospital, part of Karolinska Hospital (KS), also in Stockholm. Supervisor at NADA was Assistant Professor Örjan Ekeberg and supervisor/employer at KS was Assistant Professor Anders Fagergren. Examiner at SANS was Professor Anders Lansner. I would like to thank them all for their time and help.

I would also like to thank my co-worker, M.Sc student Jonas Natanaelsson, with whom I developed the system. Many thanks also to support technician Mikael Dahn at KS, without whose help this project would have taken additional months to complete, and thanks to all the friendly and supportive personnel down in the Motorics Lab. Thanks to Ove at Perspektiv KB for your help with the hand rig modification, and to Påvel and Johan for your constant input during the development.

Table of Contents

1 Introduction....1
1.1 Background....1
1.2 fMRI....1
1.3 Overview and Project Boundaries....2
1.4 Specification and Requirements....3
1.5 Methods and Models....5
1.6 Separation of Tasks and Limitations....6

2 Real-Time Systems....7
2.1 Real-Time....7
2.2 The Real-Time Mechatronic System....8
2.3 The Market For RTOS....8
2.4 Our Choice of RTOS....9
2.4.1 RTLinux Overview....10

3 Hardware....12
3.1 Evaluating the Available Hardware....12
3.1.1 Optical Sensor....13
3.1.2 Hand Rig....14
3.2 Choosing Additional Hardware....15

4 Computer Environment....16
4.1 Installing RTLinux....16
4.2 Development Software....17

5 System Design....18
5.1 Overall Structure....18
5.2 Patient Safety....19
5.3 Building a Test Rig....22
5.4 Real-Time Control Software....24
5.5 GUI....27
5.6 Safeguards....28
5.6.1 Software Safeguards....28
5.6.2 Hardware Safeguards....28

6 Testing & Evaluating....31
6.1 Testing the Motor....31
6.2 Test Rig Trials....32
6.2.1 Microsteps....32
6.3 Testing the Accuracy....32
6.3.1 Rotation interval....33
6.3.2 Rotating hand part measurements....33

6.3.3 Motor axle measurements 34
6.3.4 Optical Sensor measurement range....34
6.4 Patient Safety - Assessment....36
6.5 Patient Trials....38

7 Conclusions....39
7.1 Accomplished Goals....39
7.2 Future Extensibility & Unresolved Issues....39
7.3 Final Words....40

References....41

Appendix A – Short overview of a few RTOS on the market....42

1 Introduction

This chapter gives an introduction to the project and its purpose, as well as the specifications and criteria for the finished system, and the project limitations. As this thesis was done together with another student, the assignment of tasks is also listed.

1.1 Background

People with certain kinds of brain damage, e.g., those who suffer from Cerebral Palsy, often have some form of *spasticity* in their muscles. The spasticity can be of different types, ranging from mild stiffness of the muscles to uncontrollable spasms. When you bend the wrist of a normal, healthy person in an extremely rapid manner the body/brain responds with a reflex counter-motion that opposes the forced movement. This mechanism exists as a means of dampening and stopping the motion in order to protect the arm from damage that might otherwise occur. People with CP sometimes have a lowered threshold and a reflex may be triggered involuntarily at less violent motions. Those with CP often have one hand that is more ore less permanently bent inwards, towards the body. This behavior can also be found in some patients who have had a stroke.

It is of interest to medical science to do studies and find out more about this process, and what happens in a patient's brain when this reflex is triggered. The brain activity can be measured through fMRI-scans.

1.2 fMRI

The abbreviation fMRI stands for Functional Magnetic Resonance Imaging. The fMRI unit uses a strong magnetic field to measure and create an image of the brain activity of the patient. Due to the fact that the fMRI unit is operating with an extremely high generated magnetic field (usually 1.5 – 2.0 Tesla, whereas the earth's magnetic field is about 50 μT) and is also very sensitive to disturbances in the field, no magnetic objects

can be used in close proximity to the field, and no moving loops with electrical current that would induce a magnetic field of their own. This makes the implementation of any mechanical system intended to operate near a fMRI unit a bit harder. Typically, one is confined to working with materials such as wood, plastics, non-magnetic metals such as stainless steel and copper, fibre optics for signal transmission, etc., the biggest obstacle here being the exclusion of electronic components in all forms – electric motors, sensors, switches, buttons and so on.

Figure 1-1. *fMRI unit.*[1]

1.3 Overview and Project Boundaries

This thesis was done at the Motor Activity Laboratory, Department of Neuropediatrics at Astrid Lindgren Children's Hospital in Solna, Stockholm. The purpose was to create a computer-controlled mechatronic system with the capacity and tools needed primarily to do a study on CP and stroke patients, and with enough extensibility for it to be modified in the future in order for other types studies to be carried out.

The current study is meant to look at the brain patterns of a patient while his/her wrist is subjected to and bent by an external force generated by an electric motor. By varying the speed at which the wrist is bent one can generate both the slower motions that do not result in a reflex, and the faster motions that do. It is also possible to separate the kind of spastic patients that are of interest (i.e., with an increased resistance above a certain speed due to reflex motions) from those with a generally increased muscle tone, *hypertonicity*. All of this while the patient's head is located inside an fMRI unit that registers the patient's brain activity.

1 Product photo of a MAGNETOM Avanto ™ fMRI unit from Siemens Medical Solutions home page, http://www.medical.siemens.com/

Figure 1-2. *The unmodified hand rig (for left hand, here seen with a right hand placed on it). The thin wooden plate that the hand is resting on is attached to the rotating hollowed-out circle sector. The rotational axis/centre of rotation can just barely be made out behind the wrist.*

1.4 Specification and Requirements

The mechanical part of the system is to some extent a modification of an older device from the PGS project [1] that has been used in studies in the fMRI unit. The earlier device consists of a rig made from hard plastic (fig. 1-2). A patient's hand can be strapped onto a rotating plate that is then used to pivot the hand around the wrist.

The circular edge of the rotating part ensures that the force exerted by the wires will always be in a tangential direction (fig. 1-3), and hence, if the wire is pulled with a constant force, the resulting force on the rotating part will also be constant over the motion interval.

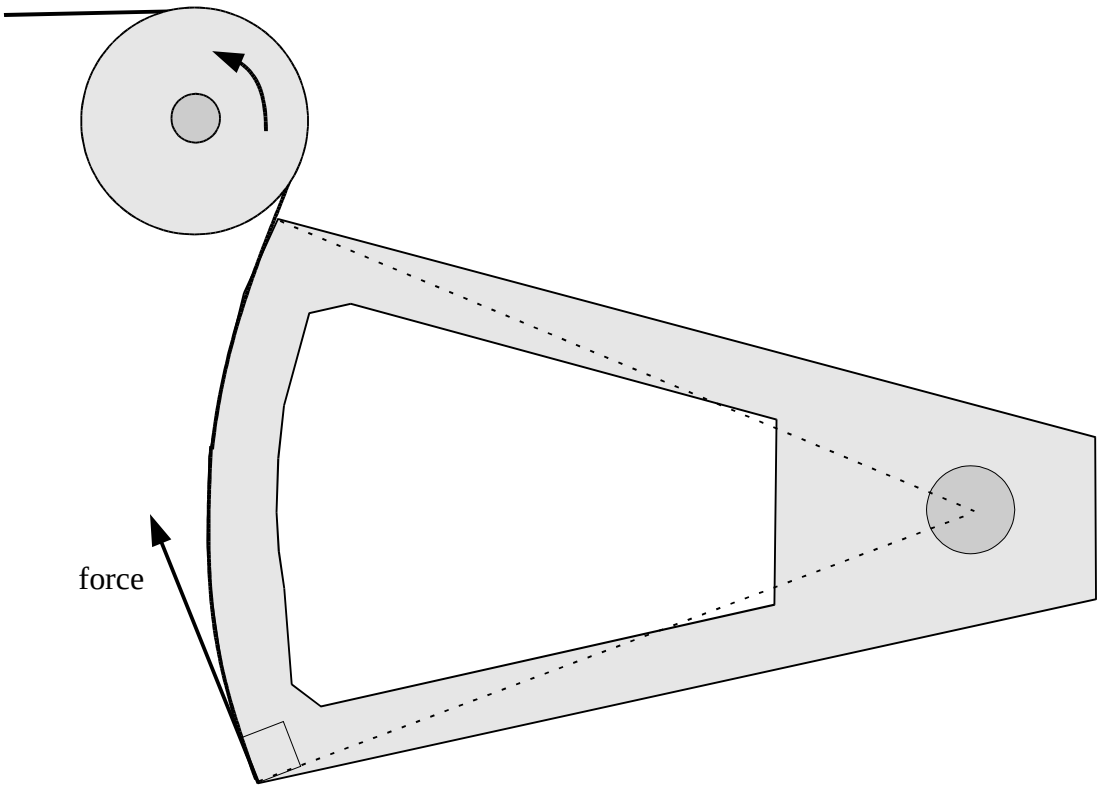


Figure 1-3. *The force vector is always tangential to the circle segment (perpendicular to the radius).*

As stated above the goal was to modify the earlier hand rig and determine which parts of the other equipment (motor, data acquisition electronics, conversion boxes) would be suitable for this new project. The final system should consist of:

- A computer, running some form of Real-Time System. Placed outside the fMRI room.
- A servomotor, controlled by the computer via an I/O card. The motor should be situated outside the fMRI room, on the floor by a row of existing holes drilled through the wall.
- The hand rig, modified as needed. The rotating part should be connected with Kevlar wires (or other material of low elasticity) running through the holes in the wall to the motor. The hand rig goes in the tunnel of the fMRI unit.
- An optical force sensor, connected via optical fibres to a Data acquisition Card (DAQ) installed in the computer. This should be built into the hand rig so that the force exerted by the patient's hand could be measured. The optical fibres also utilize the wall holes.
- A control application that allows a researcher to specify the test(s) that the patient is to be subjected to (variable speed, distance, etc).

The project was assigned a budget of 50,000 SEK (ca 6,500 USD) to cover the purchasing of the DAQ and I/O cards, plus other equipment-related expenses. The Karolinska had placed a few criteria on the final system:

- Safety first. Above all, the system should be safe for the patient. No injuries of any kind can be tolerated.
- The computer (system) must have real-time capacity. While this particular project may not suffer badly from delays, future projects for this system will undoubtedly involve responding to the patient in a much more direct manner, e.g., by constantly adjusting the motor output to match the force exerted by the patient.
- As the system will be expanded on in the future, naturally it has to be expandable.
- As for the current study, there was a requested minimum sample rate of 400 times/second with a resolution of at least 16 bits for the sampling of the optical sensor data.
- Physical characteristics of the motion of the rotating part of the hand rig were more loosely specified. It should be possible to do both fast and slow moves, where a slow move would be perhaps 5 degrees/second and a fast move could be up to 300 degrees/sec.
- When the system is built, the elasticity of the wires should be compensated for if necessary through variable motor speeds.

1.5 Methods and Models

The project was made in cooperation with another student, Jonas Nathanaelsson, a MSc student at the Mechanical Engineering programme, also at KTH. He devised the method of development that we followed, going through these steps:

1. Create a specification
2. Investigate existing equipment
3. Design a new mechatronic system
4. Obtain materials/equipment
5. Assemble the system
6. Test the system
7. Calibration and adjustments

The specification is the one given in 1.4. While JN investigated the equipment, I surveyed the RTOS market (chapter 2).

1.6 Separation of Tasks and Limitations

The separation of tasks seemed fairly natural, me being primarily responible for all things computer-related, and him responsible for most of the mechanical aspects. Quite a few tasks were also carried out collectively.

Combined efforts	
◆ Build a wooden test rig for use during the development process. ◆ Investigate possible hazards to patients. ◆ Devise safety systems to eliminate found hazards.	
Mats Uppström	***Jonas Natanaelsson***
◆ Investigate and choose a suitable RTOS. ◆ Develop real-time software for controlling the motor via one DAQ and sample data from the other. ◆ Develop a GUI for easier control of the system. ◆ Implement software safeguards. ◆ Compose a User's Manual for the control program.	◆ Investigate the motor and electronics, and decide on what equipment to buy. ◆ Modify the hand rig as needed for this new study. ◆ Build some kind of platform for the motor and electronics. ◆ Implement hardware safeguards. ◆ Compose a guide to setting up the parts of the system (wires, cables, etc).

Table 1-1. *Division of tasks*

2 Real-Time Systems

This chapter gives an overview of the concept of real-time systems as well as different subtypes. For the initial part of this thesis, a brief survey of the market for real-time systems was made. The findings are also mentioned here.

2.1 Real-Time

The meaning of real-time is not completely uncontroversial and several definitions exist, some of them contradictory. In the POSIX 1003.1 standard [2], however, real-time for operating systems is defined:

"Realtime in operating systems: the ability of the operating system to provide a required level of service in a bounded response time".

From a real-time system we therefore demand that it is able to execute some requested calculations within a certain set time limit, a deadline. Ordinary general-purpose operating systems (GPOS) do not possess this quality. They may execute fast enough on a powerful computer, and response times may be great – but there is no guarantee of consistency.

A further classification of RTS exists, in how strictly the system enforces its deadlines, and how serious its view is on missed deadlines [2]. A RTS can provide either *hard, firm* or *soft* real-time. Soft real-time is when the consequences of a missed deadline are less serious, where delays can be accepted and recovered from. Delayed calculations are still considered to have some (diminished) value, and the resulting loss of quality is deemed acceptable. If, instead, a tardy calculation is considered worthless, it is called firm real-time – the loss of quality is unacceptable. And if, in addition, a missed deadline would or could have disastrous consequences (e.g., in a fighter jet flight control system), hard real-time is needed, i.e., it must not miss a single deadline at any time.

2 POSIX (Portable Operating System Interface) is a set of IEEE standards designed to provide application portability between UNIX variants.

One should also make a difference between *desktop RTOS* and *embedded RTOS* [3]. A desktop RTOS is typically an OS running on a regular PC or desktop computer, with a screen, keyboard, mouse, etc., where extra (unnecessary) software isn't a problem (within reason), as ,e.g., a few 100 MB of Linux programs. An embedded RTOS on the other hand is more likely something you would put into a micro-controller card where the memory is limited, in a CD-player, credit card reader or some other, often commercial, application where the cost of the hardware needs to be kept down. It is also easier to analytically prove that the system is stable and correct the smaller it is.

2.2 The Real-Time Mechatronic System

Real-time and/or embedded systems used to control mechanic manipulators is not a new idea, which type of RTOS you would use depends largely on the application, available resources, etc. Examples of earlier projects that have used RTLinux are the SRA, Scanning Radar Altimeter system, developed by NASA, for analyzing hurricanes [4], and the two humanoid robots “H6” and “H7”, built by the Aircraft and Mechanical Systems Division of Kawada Industries Inc. in cooperation with Tokyo university. Examples are plentiful.

Figure 2-1. *“H7”robot, running RTLinux.*[3]

2.3 The Market For RTOS

There is presently a great number of RTOS on the market, both commercial and freeware ones – the total number is well above a hundred. Most of these, however, have very small market shares due to some being written for a specific processor family, others can be those developed by a company for internal use and later being made commercially

3 Photo from Jouhou System Kougaku Laboratory home page, http://www.jsk.t.u-tokyo.ac.jp/

available.

Choosing a suitable RTOS can be tricky enough unless you have a great deal of knowledge about the area. A survey made by the company *Dedicated Systems Experts* [5], [6] shows that customers more often than not are dissatisfied with the RTOS they are currently using, whether it is commercial, freeware or privately developed; the main reasons are that the customer's expectations of what a RTOS should be capable of varies greatly, and that documentation and support often are bad. In addition, the general quality of RTOS today is quite substandard. The company will, of course, speak well of the product they are selling, and it is only when a RTOS has been purchases and delivered that it is possible to do an in-depth examination of the system and discover how well or poorly it works.

2.4 Our Choice of RTOS

For the application we were building a soft RTS could in theory have sufficed, or at least a *weakly hard* RTS [7] since it would not be a disaster if some bits of data were lost – the patient could simply be told to repeat the move – but due to the future applications intended, we really had no choice but to go for the full hard real-time.

Among the future applications that the researchers at Karolinska have in mind is, e.g., the ability to act dynamically on the data received from the optical sensor to produce different forces acting on the patient. The original specifications for this project actually involved using this kind of feedback to simulate different types of objects, with different mass and inertia.

Before choosing a RTOS for the project a few of the most popular ones, both commercial and freeware, were investigated (see Appendix A). The RTOS that was eventually chosen is a small freeware system called Real-Time Linux or RTLinux, developed by Victor Yodaiken and others, at Finite State Machine Labs Inc. It is a modification to the Linux kernel that allows real-time processes to be run in an ordinary Linux environment.

The main motives for choosing RTLinux were first and foremost the price. RTLinux is available both as a freeware version called RTLinux/Free (this is the one we used), licensed under GPL and RTLinux Open Patent License, but also in as a commercial one, RTLinux/Pro. The free version is about 2 years old and not actively supported by FSMLabs, while the Pro version is. The price range for most commercial RTOS:es is somewhere around 4-5,000 USD, with a few going as high as 17,000. This would put an unacceptable economic strain on the project with its budget of nearly 6,500 USD as other expensive equipment was needed, plus it is good to have a little extra for unforeseen expenses.

Another important motive was the general development environment of the system. When programming, one can use the Linux environment with its standard (and thoroughly tested) editors and compilers but also other useful stuff like having access to the Internet while developing, e.g., for making it possible to search for example code

when you get stuck and – for more serious problems – the forums. This is also a benefit of the freeware status of RTLinux/Free – there are experienced users out there that can help you with your questions if you ask nicely.

It is also easy to use the real-time functionality. Programs can be written in regular Linux, in well-known languages such as C, and can be compiled and run directly without complicated procedures. User Interfaces and such things can be constructed using Linux toolkits. In short, RTLinux provides *"hard real-time, low-level environment when you need it – and all the power and flexibility of Linux when you don't."* [8].

2.4.1 RTLinux Overview

The core of any operating system is the kernel. This is the underlying process that provides the bare bones of the system. The RTLinux kernel is a modified version of an ordinary Linux kernel, that has added functionality for scheduling and executing processes that need hard real-time constraints imposed. This modified, real-time kernel works just like a standard Linux kernel, running programs, handling system resources, communicating with the hardware, etc., but unlike the Linux kernel, the RTLinux kernel overrides the regular Linux process wherever needed. Where in standard Linux, all requests and interrupts go through the kernel, RTLinux allows the user to write his/her own interrupt routines and applications that demand real-time capabilities, and can let the non-real-time parts be handled through the standard Linux channels.

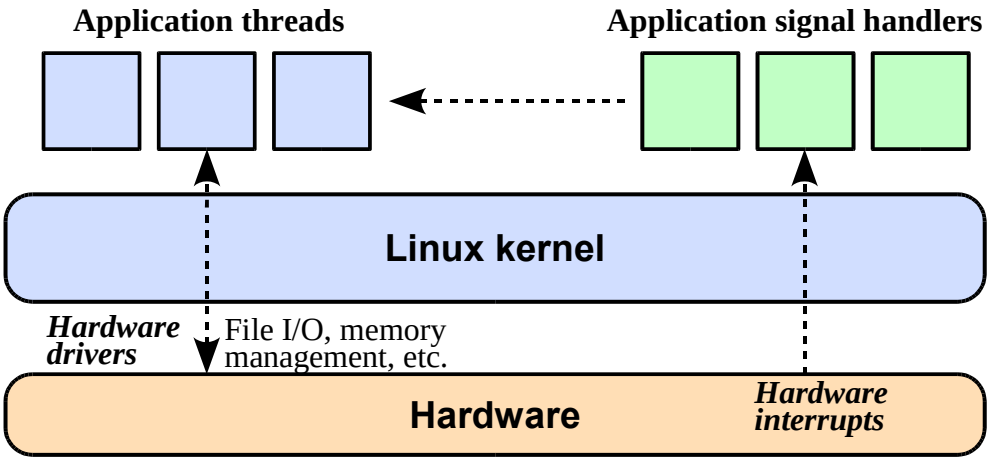


Figure 2-1. *Standard Linux system architecture. All communication with hardware goes through the Linux kernel.*

To create a real-time process in RTLinux it must be compiled as a kernel module, and then loaded with either the **`insmod`** command or with RTLinux own **`rtlinux start`** command. This module can spawn one or more real-time threads that execute with real-time constraints, and because they are threads, they will share the same memory area.

The communication between real-time and non-real-time processes is done by either of two built-in methods. The first way is to use **`mbuff`** to define an area of memory as a common buffer. Both processes can now read and write to this area. The other option – and the one I've used for this project – is through FIFOs. A real-time FIFO can be set up using **`mkfifo`** and works in the same way a normal queue would. The real-time thread can write collected data to the FIFO with real-time priority and a non-real-time process can read data from this buffer when it has time to do so.

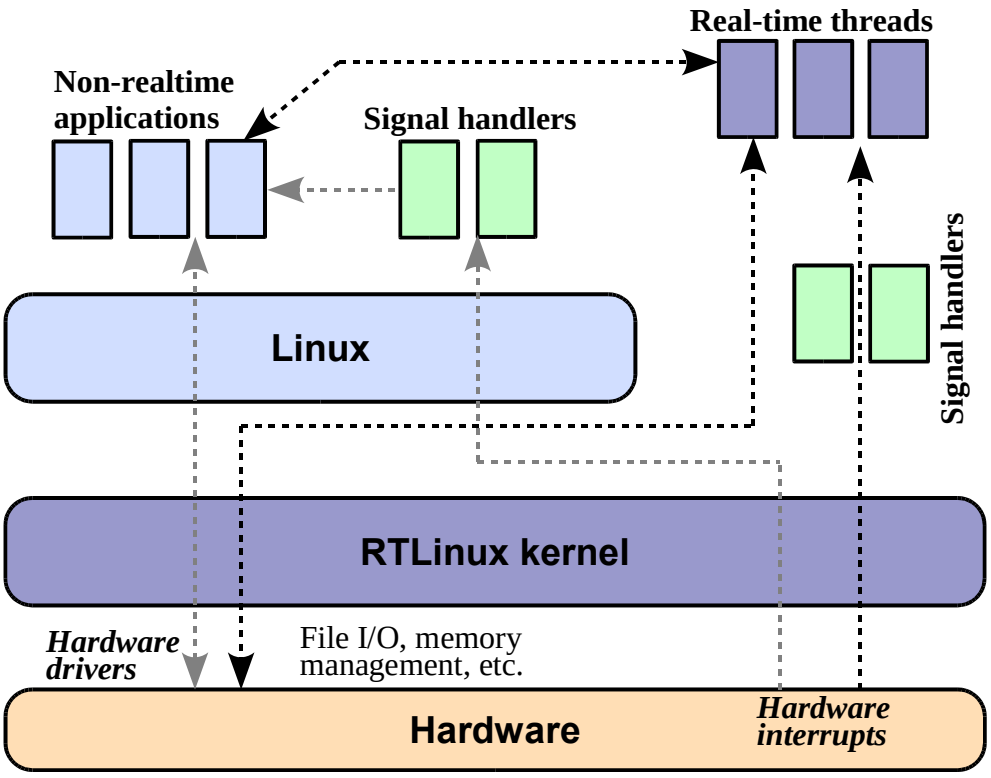


Figure 2-2. *RTLinux system architecture. Real-time threads can communicate with hardware and other processes using the shortcut provided by the RTLinux kernel, while non-real-time processes will be routed through regular Linux.*

3 Hardware

This chapter contains an overview and evaluation of the hardware initially allocated to the project, which pieces of equipment that were adequate and which were not, and the additional equipment that was purchased to replace those.

3.1 Evaluating the Available Hardware

The equipment already in the lab consisted of most of the essential pieces for the project such as an old computer (Pentium II), the hand rig from the old PGS project, an electric motor (S63-04, Lafert Automazione, not a stepper motor though), a power supply box for the motor, a custom-made optical sensor with matching electronics and a home-made couplings-box.

Looking over the equipment we decided that most of the gear was adequate for our purposes, while other things would have to be replaced.

The computer was not state-of-the-art but that was never necessary since the application would be rather simple. It was deemed good enough.

The specifications of the motor was looked at by JN and he decided it did not have the kind of functionality we needed. It was designed for constant running at high speeds, whereas we needed a motor that would move only short distances with accuracy and as constant speed as possible. We also believed that the motor would not accelerate to a given speed fast enough for our demands, and that the actual speed/force would vary during different parts of a revolution. This theory was confirmed by Mikael Hellgren, research engineer at the Institution of Machine Construction at KTH, who was the supervisor of the previous PGS project. As a replacement JN Chose a 2-phase high torque stepper motor, model ST8718, complete with a new power supply box, from a company in Germany, Solecro. When we no longer had a use for the old motor, we could also skip the old couplings-box that had been custom-build for it.

3.1.1 Optical Sensor

The optical sensor is a custom-built device, constructed by the Department of Physiology at the University of Umeå in Sweden. The idea is quite simple. A beam of light is sent through the optical fibres and through a narrow passage inside the sensor (fig. 3-2). As an external force is applied to the sensor, the passage gets even narrower and blocks some light. The amount of light that can pass through the sensor is measured by an electronics box and converted to a voltage level.

Figure 3-1. *The optical sensor, exterior view.*

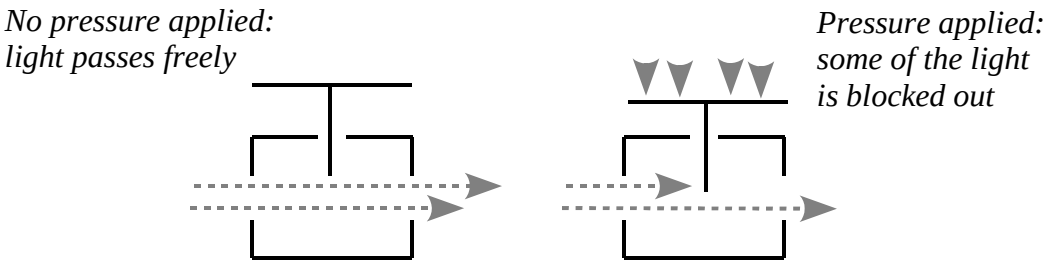


Figure 3-2. *The mechanics of the optical sensor.*

The optical sensor would also stay in the project, since it was the only method of measuring the force inside the fMRI unit. However, some quick testing showed that it was much too sensitive to be used if placed directly under the patient's hand as we had in mind initially. Measuring the sensor capacity we found it to peak at a total force of about 24 Newton (fig. 3-3), and as the hand and a part of the rig rested on the sensor, it could only measure in a very small interval before peaking. This problem would later have to be solved.

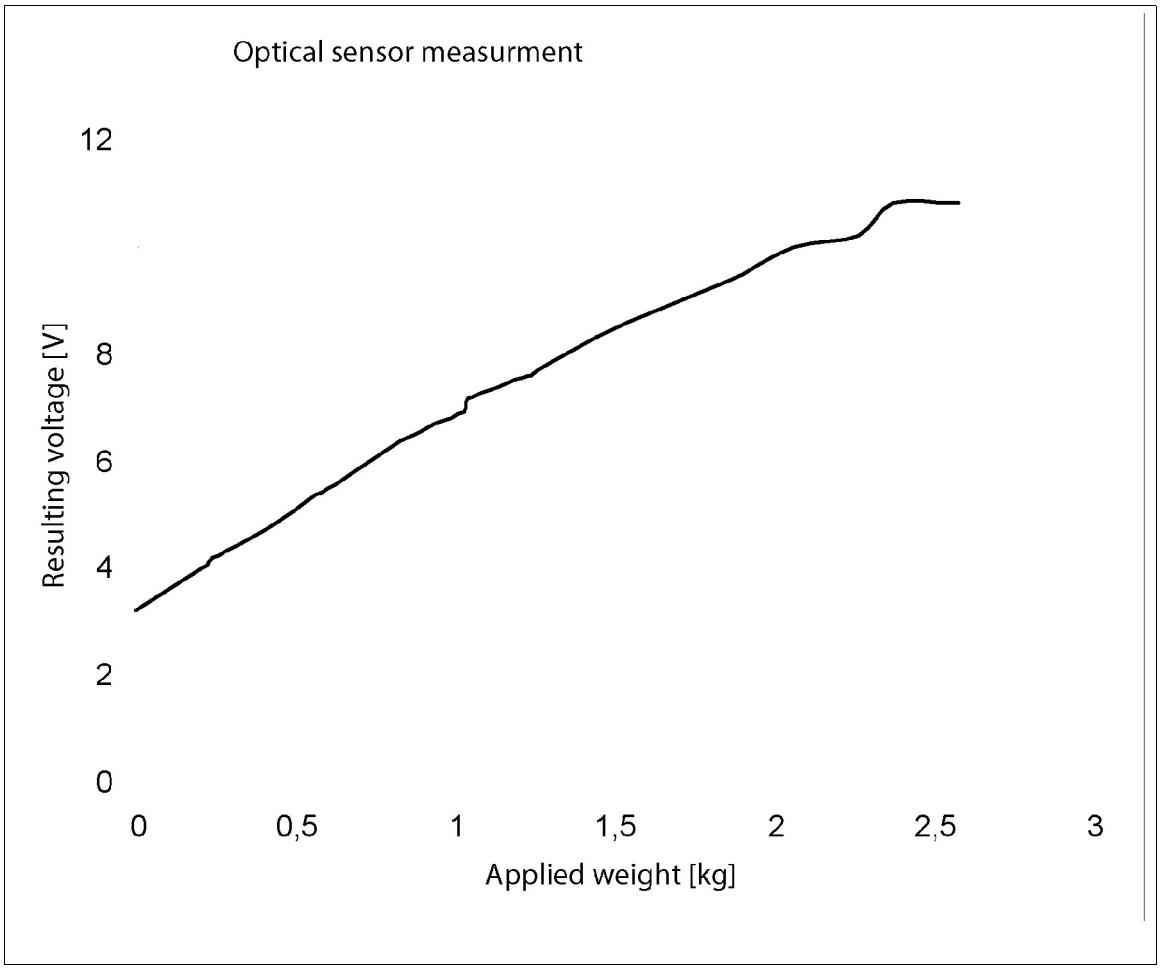


Figure 3-3. *Force-response curve of the optical sensor.*

3.1.2 Hand Rig

The hand rig from the PGS project is a sturdy thing that was designed and built to fit in the fMRI unit and had been tried out before. It was to be used in this project also, modified only in a non-disintegrating manner, so that it could be reassembled in its previous form again (still infrequently used by some studies).

The rig consists of a stable platform to which a rotating circle-segment is attached.

Also, the platform has a total of four wheels suspended by plastic ball bearings, where the wires are supposed to run. On one side of the rotating part, a plate i attached at a right angle. This is supporting the hand, so that the patient can press down and rotate the circle-sector, and conversely, the motor can pull the wire and rotate the sector, thereby pivoting the patient's hand.

An important accessory to the hand rig is the 'back-piece' (for lack of a better term). This construct is made of wood and wide enough not to fit inside the fMRI-unit. It is placed over the rear opening of the fMRI-unit and a plastic tube[4] is then used to attach the back-piece to the hand rig. The hand rig is subjected to pulling forces from the motor – the back-piece provides the necessary counter-force to keep the hand rig in place.

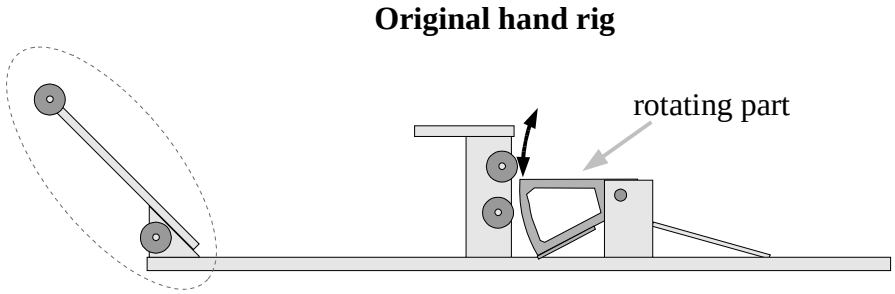


Figure 3-4. *The hand rig in its original configuration. The components within the dotted line were disassembled and removed as they were not needed for this study.*

3.2 Choosing Additional Hardware

Mainly two pieces of hardware would have to be purchased – a new stepper motor, and DAQ card(s) capable of handling the motor control and data acquisition. The exact choice of equipment was left to JN. He chose to purchase two specialized DAQ cards, APCI-1564 and APCI-3120, supplied by AddiData. The APCI-1564 card is capable of digital I/O only and was intended to send control pulses to the stepper motor. The APCI-3120 card is capable of analog I/O, for sampling the voltage level produced by the optical force sensor. He also decided to buy one position encoder for the test phase of the system.

4 Ordinary hard plastic tubing of the kind you use to protect electrical wires.

4 Computer Environment

This chapter contains a short description of the installation of RTLinux and the development software used.

4.1 Installing RTLinux

On paper, installing RTLinux looks, if not easy, then at least doable. In reality it can be both difficult and time-consuming. To install, the raw kernel source-code needs to be patched with the correct RTLinux patch, and then compiled. The patch version number must be compatible with the version of standard Linux that is to be used. RTLinux is very picky about this.

The Comedi project develops open-source drivers, tools, and libraries for data acquisition, and Comedi is a collection of drivers for a variety of common DAQ boards [9]. When the DAQs that had been ordered arrived, we also compiled and installed the device drivers (Comedi) for the cards. The Comedi drivers also needs to be of a version compatible with both the Linux and RTLinux versions.

The drivers are implemented as a core Linux kernel module providing common functionality and individual low-level driver modules. Writing our own device drivers was never really a realistic option, even though it was planned that way in the original time plan. The two DAQs used were supported by Comedi, and Comedi supports real-time card operations, necessary for the project, as it would not matter that the operating system supports real-time, if the card drivers themselves do not.

So we basically had three pieces of software whose versions had to be inter-compatible with each other:

1. The Linux kernel.
2. The real-time patch for Linux.
3. The Comedi device drivers.

Looking at the Comedi documentation, we found that it was guaranteed to support only a bit older versions of Linux. The versions used in this project are:

- Linux kernel version 2.4.4, installed as part of a RedHat 7.1 installation.
- RTLinux version 3.1.
- Comedi version 0.7.58, and Comedilib 0.7.15

After a bit of tinkering we got all three parts to work together – the Linux kernel, patched and compiled to into RTLinux, with Comedi drivers.

All in all, the installation took us over two weeks to get right. Neither of us had any prior experience with installing Linux or compiling kernels, and that certainly contributed to the amount of time we expended at this point.

4.2 Development Software

The real-time process that controls the motor and samples the data was written in C, using the XEmacs editor. The process does not communicate directly with the user, so no GUI is included in this part. The GUI instead lies in a separate program, constructed in C++ using the visual development environment Qt Designer by Trolltech. The version used is the free version released under the GPL.

5 System Design

The chapter describes the finished system in detail and the environment it was tested in. It also states the potential safety hazards to patients, and the software and hardware safeguards that were built into the system as a result.

5.1 Overall Structure

The necessary components and layout of the system dictated the kind of solutions that could be found. The main tasks were figuring out how the connection motor-wires should work, and also how to increase the effective range of the optical sensor.

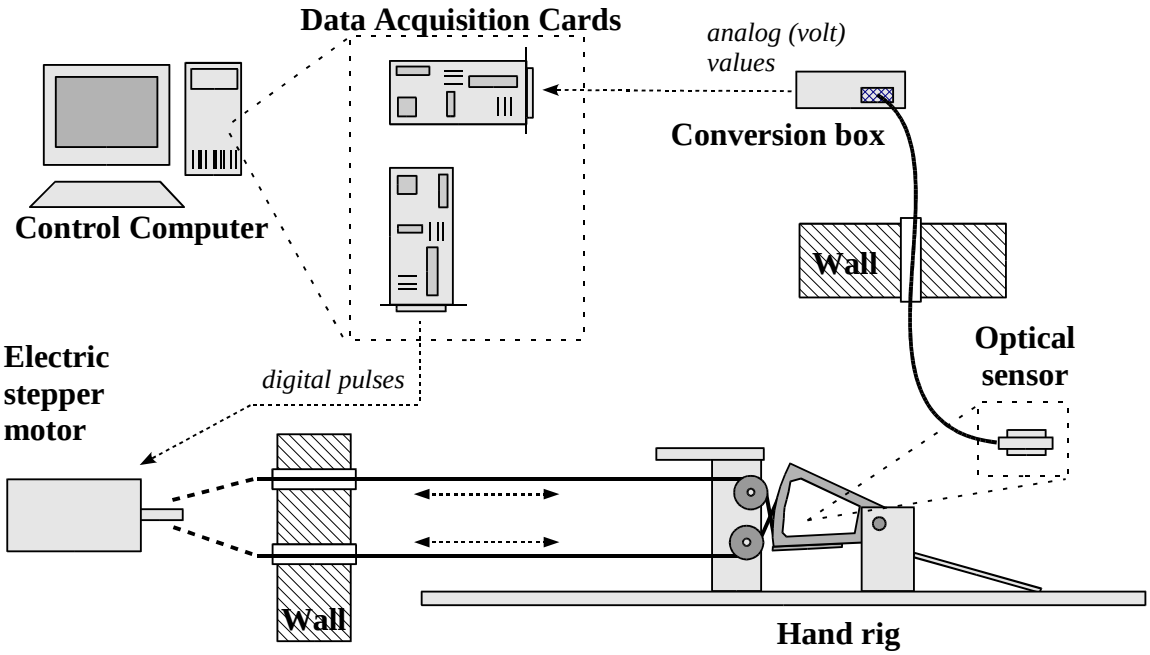


Figure 5-1. *A schematic overview of the system.*

The motor-wire connection problem was solved by prolonging the motor axle with a plastic tube and simply winding the wire around it. This coil-like structure and a piece of tape generates enough friction to keep the wire in place.

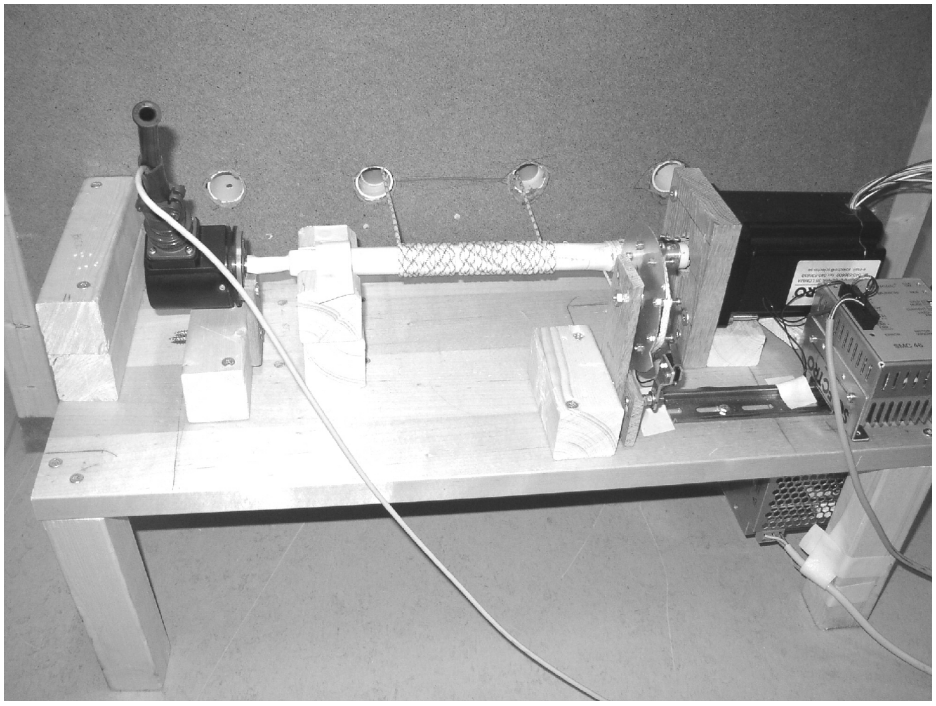


Figure 5-2. *The motor axle with wound-up wire. To the left, the digital encoder can be seen.*

The sensitivity problem with the optical sensor was harder to find a solution to, due to space considerations. When an adult person lies in that narrow tunnel inside the fMRI-unit together with the rig, there is hardly any space left. We finally came up with a design that we could believe in. Due to the technical nature and need for robust implementation of this design, the task was contracted to the same mechanical engineer (Ove at Perspektiv Konstruktionsbyrå) who had built the original hand rig years before. The simple idea behind the design is to let the edge of non-magnetic metal plate (copper) bear some of the weight. This design is described in greater detail in Natanaelssons thesis [10] (pg. 22).

5.2 Patient Safety

The safety of the patient was the part of the specification that our employers at Astrid Lindgren's described as the most important criteria of the system. Under no circumstances should the patient be injured. In order to assess the safety of the system, we must first identify all the risks, the potential damage that might be inflicted on the patient. I have used the process suggested by G. Bartoo [11], that follows the AAMI/ISO 14971 –

standard for risk management.

This led me and JN to sit down and have a small brainstorm on the possible hazards. I have chosen to identify the hazards from the patient's perspective, i.e., by focusing on the way he/she can be injured irrespective of which part of the system that causes this. We found 5 categories of possible injuries:

1. **Bending too far.** The system injures the patients hand by bending it too far, beyond the natural limits of the joint.
2. **Bending too harshly.** The hand is injured by forcing a powerful move on a very stiff joint or one with reduced flexibility.
3. **Changing direction rapidly.** A malfunction causes the motor to jerk the hand back and forth rapidly, possibly injuring the hand.
4. **Pinching the hand.** Since the hand rig has moving parts as well as stationary, the patient's hand could get stuck between them and injured seriously. The motor is strong.
5. **Scratching or bruising.** Other injuries not related to the hand-motor interaction, like the patient bumping into the hand rig and perhaps getting a bruise.

These were kept in mind during the design of the system. It is also important to remember that the system will not hurt the patient through inactivity, like a pacemaker would. If it breaks down and stops, then the patient is at least safe. This means that only malfunctions that cause the system to act on the patient need to be considered in the safety analysis. If the wires snap or the caster wheels break off, that will only make the wires go limp and unable to affect the patient any further (not good of course, but at least safe).

The probabilities of there hazards occurring were evaluated, and in some cases we even found it impossible for the conceived event ever to take place due to the construction of the hand rig. Safeguards were built into the system for those risks that could not be ruled out. The estimated probabilities of the hazards can be found in chapter 6.4.

Continuing with the risk management procedure outlined in Bartoo's article, I created simple tables for the severity and probability criteria of the possible hazards. Table 5-1 lists the severity of hazards. Those that may result in bodily injuries I have rated as being of high severity, the others as low. Table 5-2 lists the different probability levels that a hazard may have.

Severity	Description
Major	*Could result in serious physical injury to the patient and/or operator.*
Minor	*Not expected to result in any injury to the patient or at most a possible non-serious injury.*

Table 5-1. *Severity criteria*

Probability of occurrence	Description
Frequent	*Likely to occur at least once/month in the operating life of the system.*
Probable	*Likely to occur less than 12 times/year in the operating life of the system.*
Occasional	*Likely to occur at least once in the life of the system.*
Remote	*May occur in the life of the system.*
Improbable/Impossible	*Unlikely to occur in the life of the system.*

Table 5-2. *Probability criteria*

Now we need to determine a risk level lookup table that can be used to decide if a hazard is acceptable or not. The probability of a hazard is crossed with its severity and a final verdict is reached. I have chosen to simplify this scheme down to only two risk levels.

One of the goals of this project is to make sure that the mechanical system will never injure the patient. Therefore a high severity hazard can only be acceptable if the probability of this hazard occurring is extremely unlikely or even nonexistent. The low severity hazards I put as acceptable unless they are frequently occurring. This table pertains to this project only – different projects can have their own tables.

		Hazard severity	
		Major	**Minor**
Probability	**Frequent**	*Level 2 – unacceptable*	*Level 2 – unacceptable*
	Probable	*Level 2 – unacceptable*	*Level 1 – OK*
	Occasional	*Level 2 – unacceptable*	*Level 1 – OK*
	Remote	*Level 2 – unacceptable*	*Level 1 – OK*
	Improb./Impos.	*Level 1 – OK*	*Level 1 – OK*

Table 5-3. *Risk level lookup table*

Before the system can be put into use, all weighted risk levels must be at level 1. Therefore, all hazards with a risk level of 2 must be reduced. This can be done by reducing the probabilities of such hazards occurring or, possibly, by reducing the negative effects of such a hazard to a lower severity level, all through adding different safety features. The safeguards that we put in the system as a result are found later in this chapter (5.6). An assessment of the finished system and its potential risks, is not made here, but instead in chapter 6, which covers testing and evaluating the system.

5.3 Building a Test Rig

We quickly realized we would need some kind of artificial testing environment. Being able to test the system back in the lab would save us from many unnecessary trips to the fMRI units.

To this end we used two old office desks to simulate the fMRI bed, with thick sheets of foam rubber on top for patient comfort. We were able to attach the back-piece to the farther short end of the table, so no special measures had to be taken with this part. The wall of the fMRI room, however, would have to be simulated in some way. To this effect, we had to manufacture a wall section of the correct thickness, complete with holes, but also some kind of support structure that could hold the wall and tables in place (at a set distance from each other), since neither the wall section nor the table could be fixed to the floor. The building of the test rig was one of the first hardware related tasks, and it took around 3 days to complete, including purchasing building materials.

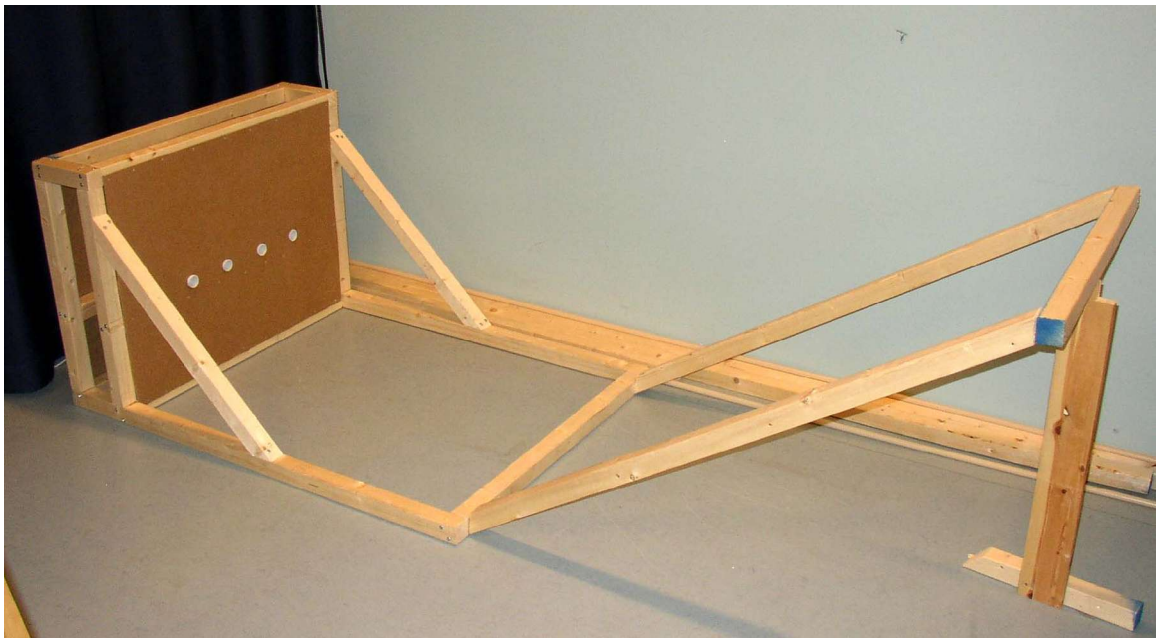


Figure 5-3. *The test rig, consisting of a wall section with holes (left) and supporting struts to keep the wall and table separated when the wires are stretched.*

JN later built a bench for the motor and some of the associated electronics. It is to be placed outside the fMRI room, up against the wall.

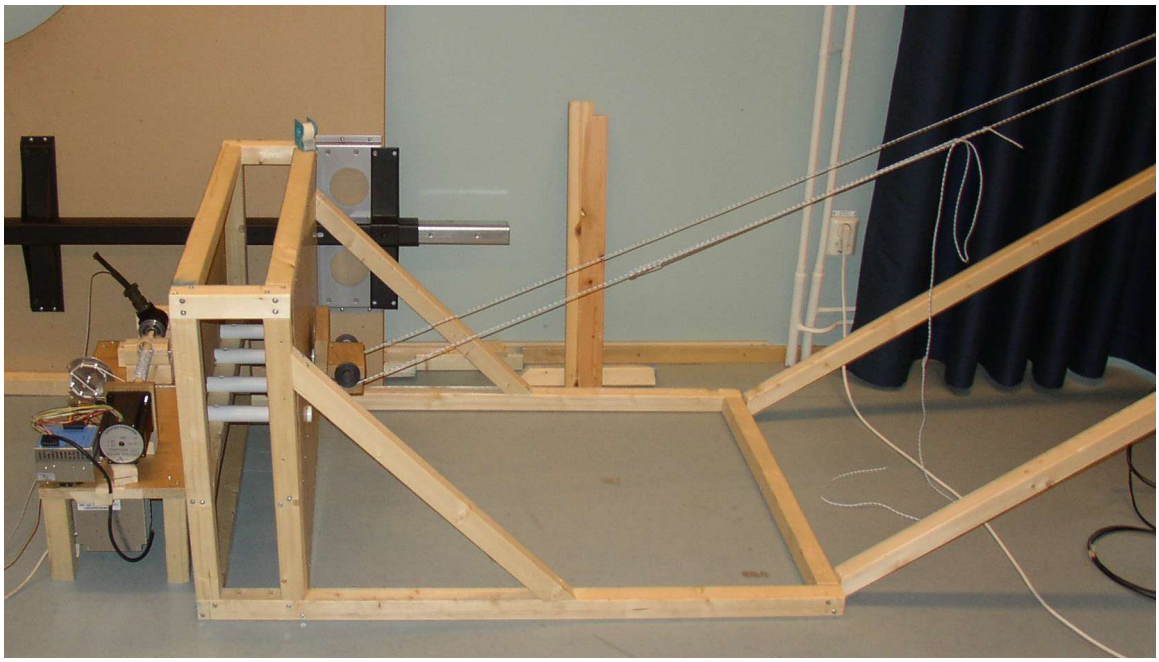
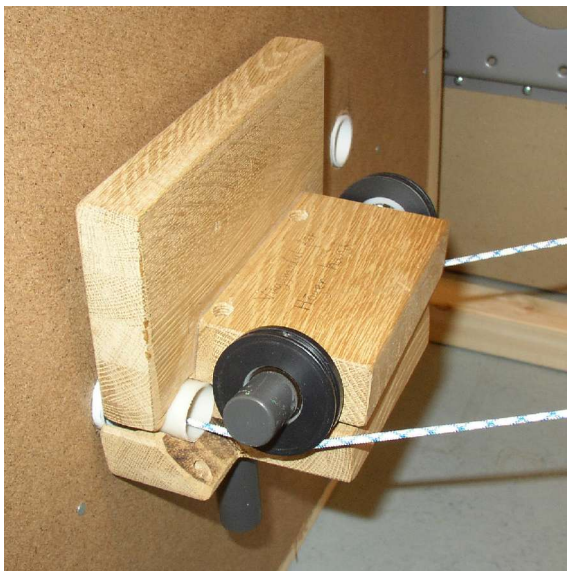


Figure 5-4. *The fake wall section with motor bench and wires set up. The sailor's knot used for tightening the wires can be seen on the lower wire.*



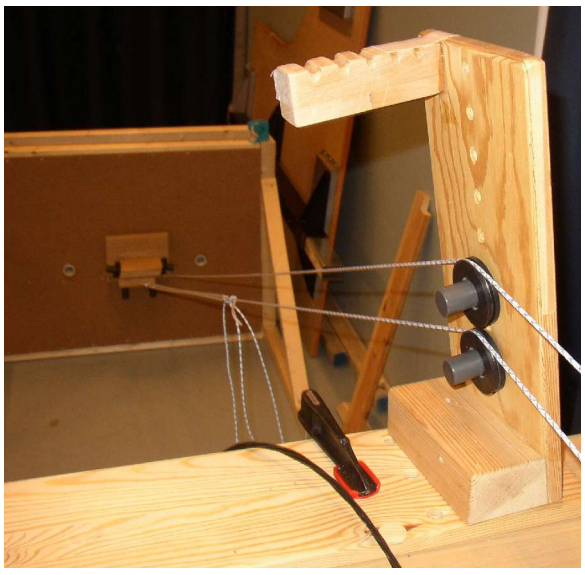


Figure 5-5 and 5-6. *Two of the existing parts from the older project.*

One problem was how to tighten the wires. Ideally, one would like some sort of simple mechanical device, like when you strap something to the roof of a car, but adding a large chunk of metal to the wire will both weigh it down and give an extra unwanted inertia to the system. The simplest and least demanding solution we could find was with a special kind of sailor's knot, creating kind of a block and tackle system (see Natanaelssons thesis for details on this).

The tighter the wires are, the quicker the hand rig will respond to the motor, but the tighter the wires are, the more damaging it will be on the caster wheels, since they have ball bearings made from small plastic marbles that will be very difficult to find replacements for once worn out. Therefore, the tighter the wire, the greater the force and the abrasion/attrition. The wires should hence be tightened 'just enough'. For this specific study it is not terribly important to have maximum tension in the wires, since we pull only in one direction at a time, and if the wire is not tight, it will be tightened after only a few milliseconds of pulling. This will only result in a delay of the entire motion, and has no impact on this particular study.

If instead motor output is to be based on sensor input, then something might have to be done about this delay.

5.4 Real-Time Control Software

The design of the system is not that complicated. There are only two processes, the real-time process and the GUI, with totally different areas of responsibility (fig. 5-8). The GUI only tells the real-time process what kind of motion it should execute, while the rt-thread handles the low-level instructions to send and receive signals through the card ports.

Our software was designed to meet the needs of the first study to be made with this new real-time system. It needed to do the following:

- Subject the patient's hand to a forced rotation, angle and speed predetermined.
- Measure the counter-force exerted by the patient.

Therefore, I designed a GUI for specifying motions to execute, and a real-time process to handle the actual execution. In this system, a motion consists of three separate phases, the first one being a kind of initialization move.

Due to the fact that we cannot measure the actual position of the rotation hand part, its position should always be at the lowest, horizontal level (the hand itself will actually be bent slightly downwards in this position, perhaps 10-15 degrees) when starting a new test. Since the GUI allows the user to specify an arbitrary interval of angles to rotate between (within the possible movement range of the hand rig, of course), there may be a need for

an initial rotation, up to the starting angle. If an initial movement of this kind is necessary, it occurs at a fixed, semi-slow speed.

The second phase is the main motion phase. This is where the requested motion is executed as specified in the GUI, within that interval, with the chosen speed.

The third and final phase of the motion, is a reset phase, where the hand is (again, slowly and with a set speed) rotated back down to the horizontal start position. Once this is completed, the system is ready to execute another motion.

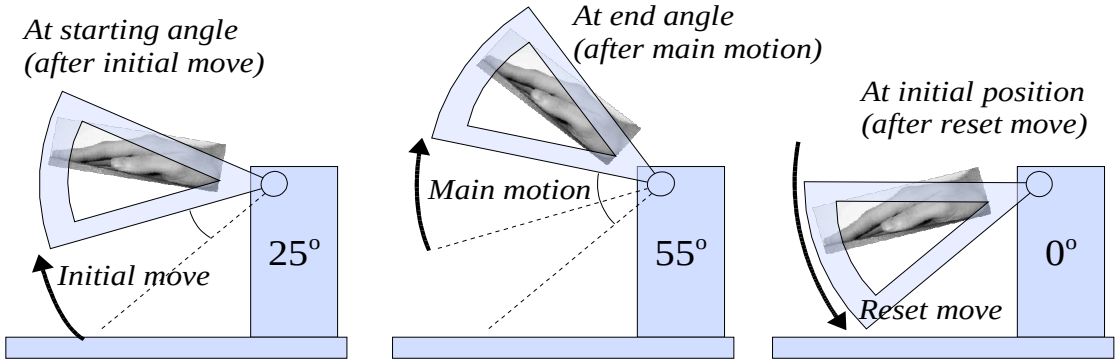


Figure 5-7. *The different phases of a motion. In this particular example, an initial move is executed since the starting angle (25º) is greater that the angle at the initial position (0º).*

When the real-time thread is not busy executing a motion it sleeps, waiting for an order to be placed in the Control FIFO. While executing an order, it continuously collects data and pushes it into the Data FIFO. The GUI is waiting for data to appear in this queue and will retrieve it when it has time to do so – the current Data FIFO is dimensioned to easily hold all data from a whole motion, so it is not critical. To signal the end of the ordered motion, the rt-thread sends an end-of-dataset marker to the FIFO when it is done. This tells the GUI to stop listening for data, save all that it has read so far to a data file. If more than one motion was specified, it would pause briefly before repeating this procedure for the next motion.

When a data file is saved, the data is plotted on the screen using GNUplot. This was implemented as a debugging tool during development but in the end we decided to keep the feature.

The researchers will want to move a copy of the data files to their home server accounts, for studying it closer. This can be done with the linux shell command **`scp`**, to securely copy a file over the network. The computer has a working network connection so this is not a problem.

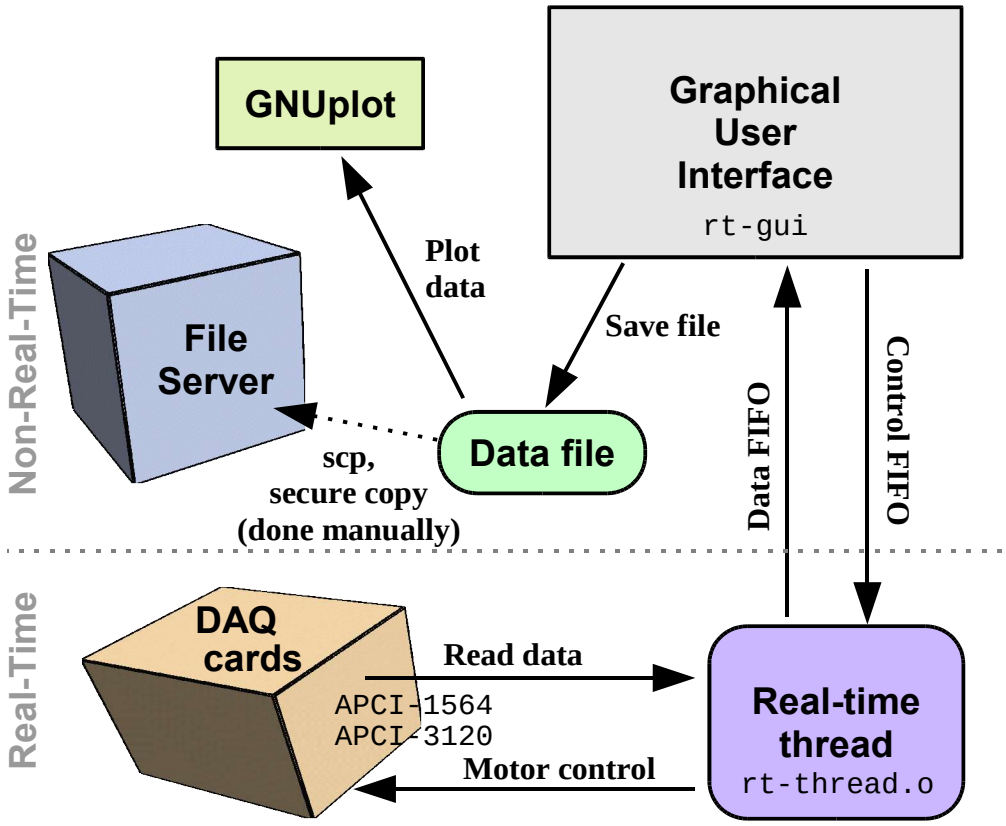


Figure 5-8. *Overview of the computer system.*

Our real-time module is called **`rt-thread.o`**. When it is loaded it spawns a thread that runs at a set frequency of 6 kHz. For this project we used a fixed cycle period for our real-time loop.

The thread then performs its operations after each set number of cycles. For example, by telling the stepper motor to take a step every 2, 3 or 6 cycles, the thread would cause the motor to move with frequencies 3000, 2000 and 1000 steps/second respectively. The sampling frequency can be set to the same or a different frequency. The reason for having both the motor control and sampling tasks in the same thread is that we wish to avoid any unnecessary scheduling problems, when otherwise two tasks could be scheduled to run at the same time with one being delayed as a result.

The problem with having a fixed cycle period is of course that it limits the number of possible motor speeds, as dividing 6000 evenly is the criterium. Also, the higher you go in the spectrum of speeds, the further apart the valid speeds are. For example, highest speed implemented is 236 deg/sec – the closest lower speed is around 170 deg/sec. In the lower end, the speeds are densely packed – e.g. both 5 and 6 deg/sec are valid speeds.

In future developments, it is probably desirable to move to a dynamically set cycle

period with motor activation every cycle, so that the speed can be actuated by adjusting the period during execution. However, then there would be a need for two separate threads for motor control and sampling, as it would not be acceptable for the sampling frequency to change when the motor speed does.

The motor can be moved at most every other cycle. This is due to the electrical-mechanical components involved. The motor is triggered on a negative flank, i.e., when the digital input signal changes from 1 to 0. Unfortunately, at these speeds we are at the limits of the operational capabilities of the electronics involved. The signal cannot be given two different values within the same real-time cycle, as a certain time is needed for the output value to settle. The system can therefore at most flip the signal between 1 and 0 once every cycle.

5.5 GUI

No conscious attempt at following any HCI-oriented design method was made during the development of the user interface. The amount of functionality provided by the GUI is rather small for this particular application, and the number of users is also small (two researchers as of now, may possibly include more later).

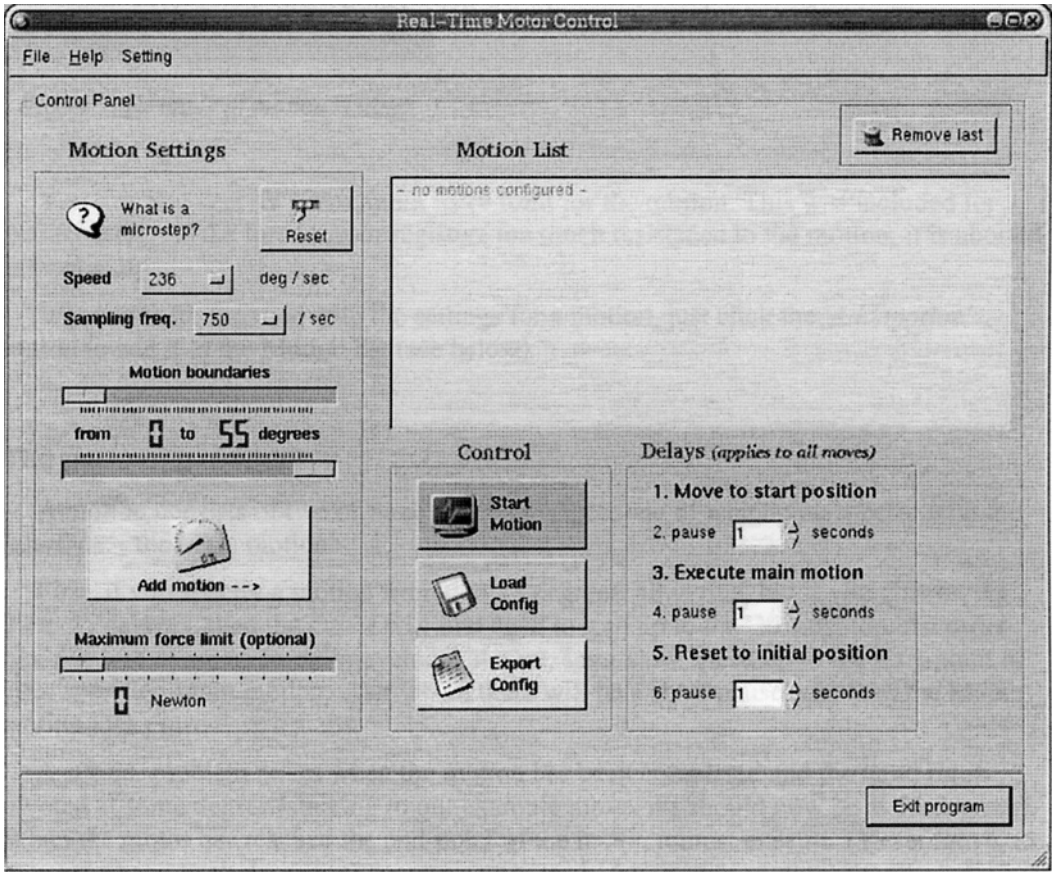


Figure 5-9. *The graphical user interface.*

A prototype GUI was initially created, and since it received some good remarks, I chose to use it as the official GUI of the RT Motion Tester. On several occasions during the development, the current state of the project was demonstrated to the intended users, and they were able to make comments about the interface - a few changes were also made due to this input, although the general structure of the GUI was good enough to remain unchanged.

5.6 Safeguards

Both software and hardware safeguards were added to the system.

5.6.1 Software Safeguards

The software contains a few safeguards for preventing an error in the program or computer to be perpetuated to the physical world. Illegal values are not permitted to be entered as parameters for the motions, and the real-time thread also checks the number of pulses it receives from the digital encoder connected to the motor axle, for signs of having moved too far.

The other software safeguard is utilising the signals from the optical sensor. If the researcher so wishes, a force limit can be specified for the motion. If the measured force exceeds this threshold, the move is aborted immediately and the hand is gently moved back to the start position.

Other that the encoder and the optical sensor, there are no other means of feedback from the real world to the computer. This limits the safety measures that can be implemented in the software part.

5.6.2 Hardware Safeguards

The motor axle is connected via wires to a larger diameter wheel (fig. 5-10). As the axle rotates, so does the wheel, so the position of the wheel depends directly on the position of the axle. If the wheel rotates too far in either direction, it will hit one of two hardware switches mounted on the motor bench. When this happens, the current to the 'enable' channel on the motor is cut, stopping the motor regardless of what other signals are sent to it. After a hardware break of this kind, the position of the motor axle must be reset by hand since the control software can no longer move it.

Also, the same loop goes through an emergency stop button (fig. 5-11), which can also be pressed to break the current. This might not be terribly useful but gives the researcher a feeling of being 'in control'.

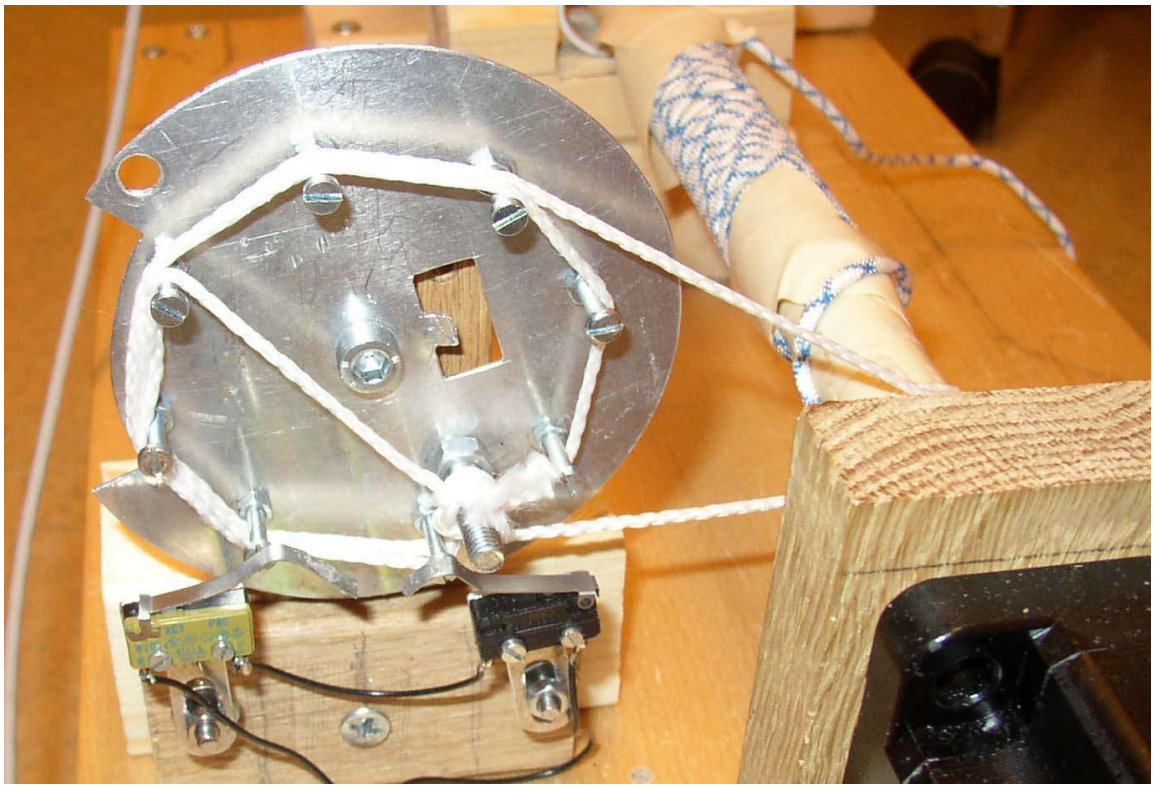


Figure 5-10. *When the protruding screw on the wheel hits either of the two end-point switches, the loop of current is cut, stopping the motor.*

Figure 5-11. *The emergency stop button.*

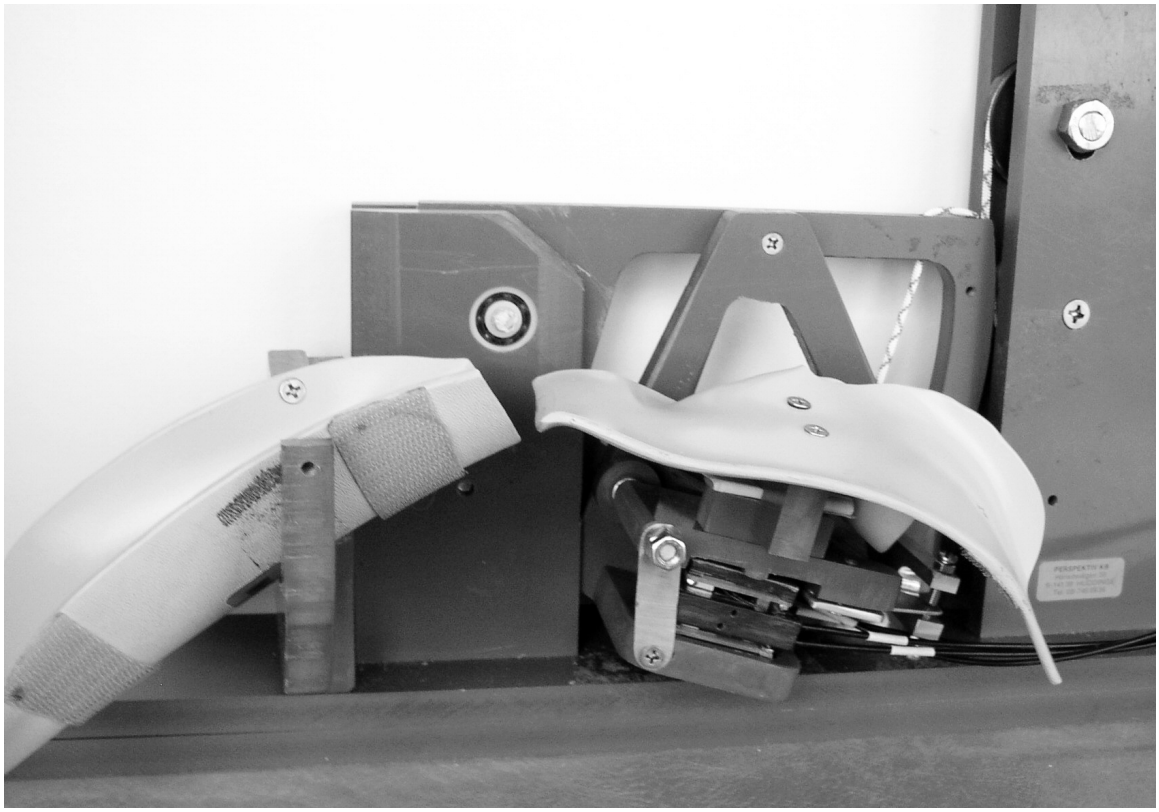


Figure 5-12. *The hand rig in its final state. The hand-shaped orthosis is mounted on a plastic slab that is hinged on one edge (to the left in the picture). The opposite edge has a flexing copper plate that absorbs some of the force that the hand exerts. The remaining force affects the optical sensor, placed in a square recess between the two plastic plates.*

6 Testing & Evaluating

This chapter outlines the different tests and measurements and the results thereof that were made during the development, up to and including a final test in the actual fMRI-unit environment. It also contains a final assessment of the safety of the system.

6.1 Testing the Motor

When the real-time control software was nearly completed we tested the motor with different speeds to see if it all would work as planned. The first thing we noticed was that when one of the higher speeds was chosen, the motor would not run. This was due to our naïve solution – the software simply instructed the motor to start running at the chosen speed, ignoring any inertia and need to gradually bring it up to the desired speed. The result was a buzzing sound and the motor axle would just vibrate back and forth.

The solution to this was to implement a short acceleration stretch for the highest speeds (also decelerating when stopping). The motor would start at a lower speed and work its way up to the target speed. The amount of time spent on each speed was determined by trial-and-error, trying to minimize the total time before reaching the target, while at the same time making sure that no motor pulses were lost due to too rapid acceleration.

The slower speeds ran at perfect precision, so by trying the fast speed with startup, and then rewinding the axle at a slow speed, the precision in the faster motion could be measured by checking if the axle was brought back to the exact same position or not. This is an important requirement for our system as it will execute many motions during a single trial.

However, no suitable acceleration stretch could be found for the fastest of the speeds, when the motor is moved every other cycle, i.e., 3000 times/sec. The move would produce inconsistent results – sometimes the motor would run too far, and sometimes not far enough, even with long acceleration and deceleration stretches. This speed was thus deemed to be too unreliable for use. Likely, this error comes from the motor failing to

handle 3000 steps per second reliably (it exceeds the limits stated by the retailer).

6.2 Test Rig Trials

Having made sure that the motor works as intended on its own, the time came to try the whole system in the test rig. The system was tested for slow as well as fast motions, with resistance and without (with one of us lying down with his hand strapped on, just like the patient would have). The motor is very strong and not even when we resisted the move at the top of our strength did it affect the motor, as fas as we could notice.

We did find a problem with one of the slower speeds. It turned out to produce some sort of strange fluctuation. The wires would oscillate wildly, almost jumping off the guiding wheels, and the motor would stutter. It is unclear if this is due to the motor having some sort of glitch at this one speed, or if it happens to correspond to the natural harmonics of the wires, or maybe a combination of both. This speed was excluded as a choice for possible parameters. There are other speeds close to it and it will not be missed greatly.

6.2.1 Microsteps

At higher speeds the steps of the motor blend into each other to create a seemingly smooth motion. As the speed drops though, the discrete steps can be clearly felt as a vibration. To lessen this feeling, the motor can be run in microstep mode. In this mode, the motor takes smaller steps, only a 10^{th} of the normal length. By taking these steps 10 times as often, one can recreate the same motion, but with a distinctly smoother feel to it. Since this means sending signals to the motor 10 times as often, this cannot be used for the higher speeds. To make things worse, you cannot switch from normal to microstep-mode by sending signals only. You have to physically set three switches on the motor itself and reset it, by cutting the power briefly. This means some extra work when both microstep and 'macrostep' motions are to be run in the same series of motions, since this procedure requires perhaps 15-20 seconds to complete (or less if you are used to doing it). Pauses between the motions can be set up in the software to allow for this procedure.

6.3 Testing the Accuracy

Due to the motor being a stepper motor, it is extremely accurate providing the correct number of steps are taken. The resolution is 200 steps/rev so each step is 1.8 degrees. The diameter of the motor axle is many times smaller than the diameter of the rotating circle segment on the hand rig, the transmission is 15.23 to 1. This uneven number also causes the speeds to be uneven, like 236 or 171 degrees/sec.

6.3.1 Rotation interval

The rotating part of the hand rig was looked at a bit closer and the maximum range of the rotation was determined. With a small margin on both ends, we found that it could safely be rotated within an interval of 55 degrees.

6.3.2 Rotating hand part measurements

The digital encoder was mounted on to the axis of the rotating part of the hand rig. This enables the rotation to be measured with some accuracy. The resulting data was fed back into and logged by the computer. As can be seen in the figures below, there was a slight delay caused by the elasticity of the wires, greatest at the highest speeds. The graph in figure 6-1 is from the highest speed that the motor can handle, i.e., 236 deg/sec at the hand rig, or roughly 10 revolutions/sec at the motor axis. Note that the full motion (almost 50 deg.) takes only just over 1/5th of a second.

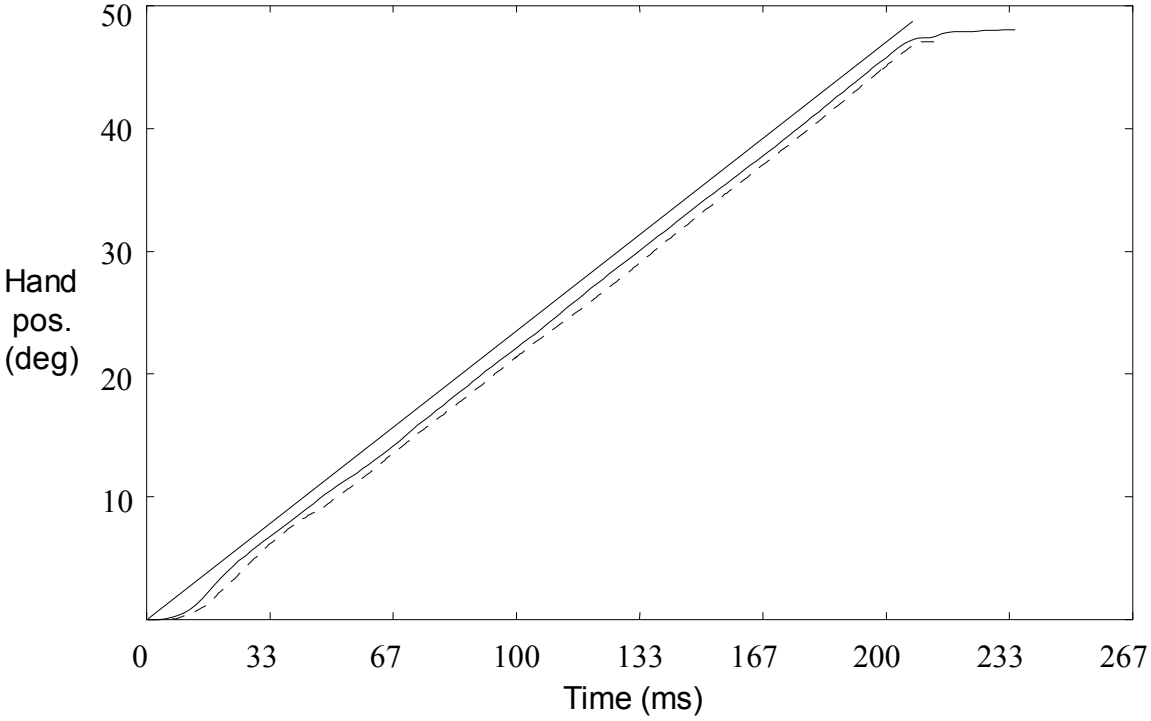


Figure 6-1. *The straight line corresponds to the optimum correlation between motor axle rotation and that of the rotating hand rig part. Note though that due to the acceleration part of the motion, it can never completely match this line. The two curves are the measured positions when executing a high-speed motion, corresponding to 236 degrees/second at the hand side. The solid line is from when the hand is limp, without much resistance, and the dashed line is from when pushing back as much as possible [around 80 N]. Encoder sampling frequency was 600 Hz.*

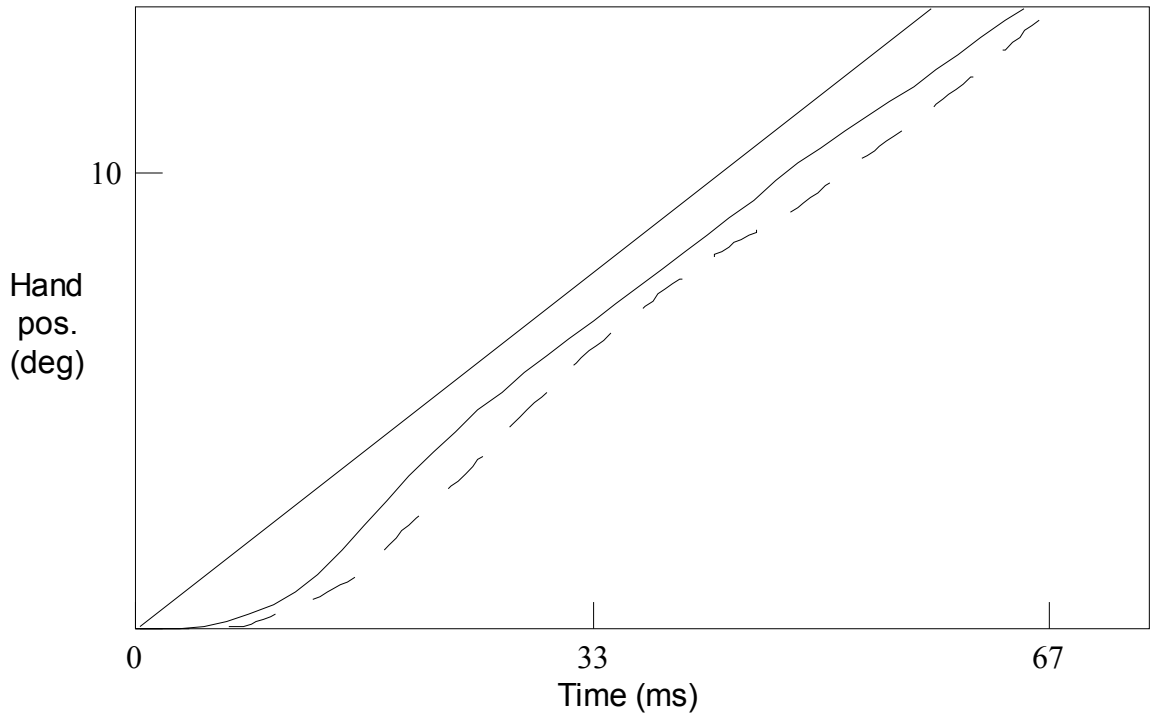


Figure 6-2. *A closeup of the curves in Figure 6-1.*

6.3.3 Motor axle measurements

Attaching the encoder to the motor axle enabled us to measure the actual speed of the motor. It came as no big surprise that the ideal and actual curves were practically identical.

6.3.4 Optical Sensor measurement range

The optical sensor itself peaks at 24 N, but the modified hand rig now takes part of the load off the sensor enabling it to measure a gross total of up to 75 N applied by the hand. Unfortunately, the transfer function is not entirely linear. To some extent this non-linearity may stem from the sensor itself, but the largest part is likely to come from the mechanical solution with its flexing metal plate.

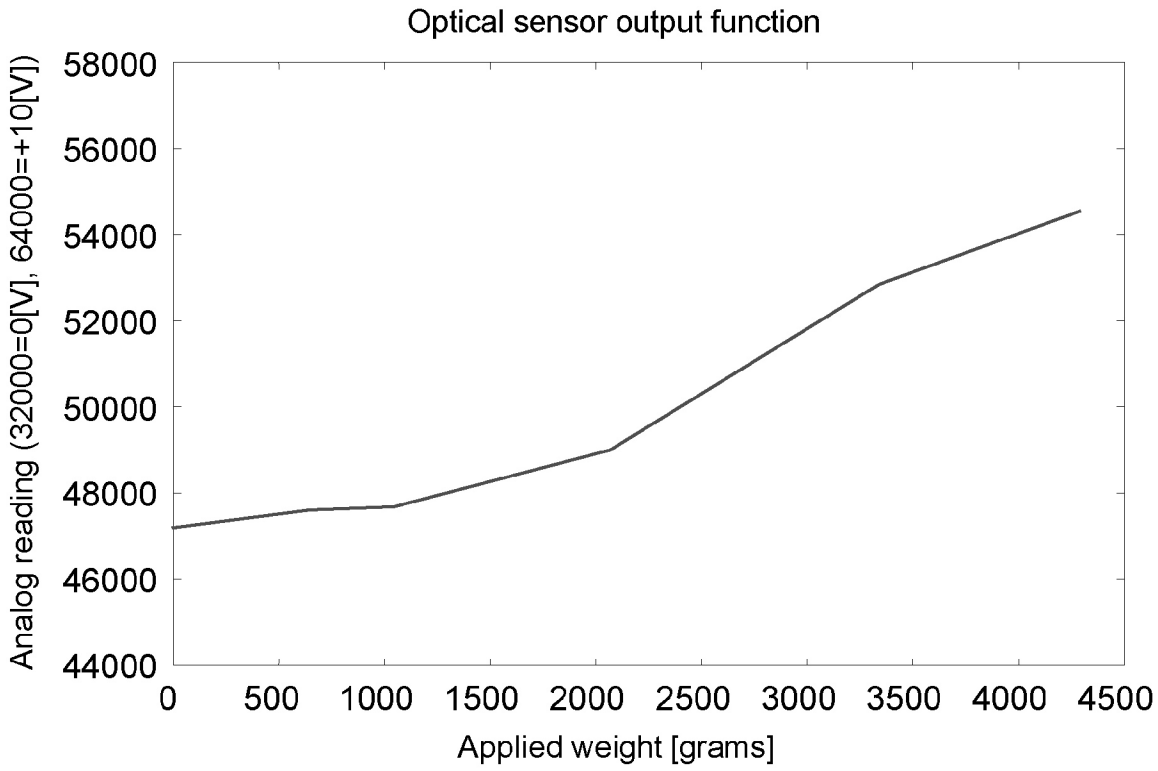


Figure 6-3. *The non-linear force-output function of the optical sensor when inserted into the rebuilt hand rig.*

In order to transform the measured analog values into their corresponding weights, a transfer function was needed. Since the curve looked rather smooth, we used GNUplot to find a function for the voltage, V, as a function of weight, w, by manually adjusting the parameters until we had an approximate match of the curve. This did not take long. The first part of the curve has the look of a second degree polynomial, and the second part more resembles a straight line (first degree polynomial).

The approximated transfer functions were:

$$V_1 = 0.000484\,w^2 + 47200 \qquad , w < 2340$$

$$V_2 = 2.5\,w + 44000 \qquad , w \geqslant 2340$$

Or inversely, in the form we want them:

$$w_1 = \sqrt{\frac{V - 47200}{0.000484}} \qquad , V < 49850$$

$$w_2 = \frac{V - 44000}{2.5} \qquad , V \geqslant 49850$$

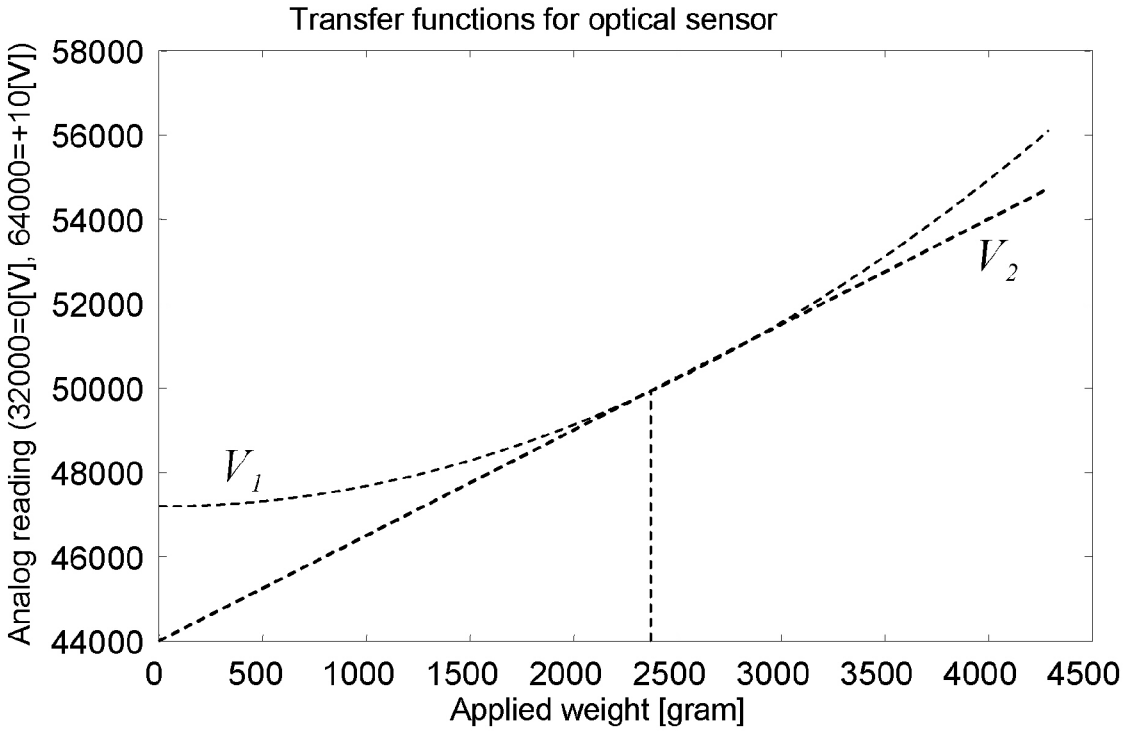


Figure 6-4. *The approximated transfer functions V_1 and V_2. V_1 is used for values under 49,850 and V_2 is used for greater values.*

6.4 Patient Safety - Assessment

Now a final look at the possible hazards from chapter 5.

Bending the hand too far.

This hazard seemed plausible at the start of the project. However, after carefully investigating the hand rig, we found that due to the layout of the rig, the wires can only convey force to the rotating part in a 55 degree interval. This hazard cannot physically happen.

Probability – *Improbable/Impossible*.
Severity – *Major*.

Forcing a stiff joint too harshly

The probability of this happening is dependent on the patient. If the patient has stiff enough joints, this WILL probably happen as soon as you run a fast motion trial. If on the other hand the patient lacks this stiffness, it will never happen. This risk forced us to implement the force limit safeguard that halts a move if the patient resist too much. The force limit must be specified in the motion parameters to be enabled (default: off). The use and parameters of this safety feature is up to the researchers themselves, since they have the best knowledge of the human body and its limitations.

Probability – *Frequent (worst-case)*.
Severity – *Major*.

Jerking the hand back and forth rapidly

The way the software was developed, the computer cannot cause this to happen. The direction the motor runs is determined by a single channel to the motor that can be either 0 or 1 and is set by the software every time the motor takes a step. The faster speeds also requires a short acceleration stretch or the motor will just vibrate and emit a buzzing sound. After considering the speeds and accelerations closer, we think it is highly unlikely that even the highest speeds, with sudden reversals, for extended periods could harm the patient's hand. The maximum speed just isn't that great.

Probability – *Improbable/Impossible*.
Severity – *Minor*.

Hand stuck between moving parts

This could possibly happen with the original hand rig. The final design though, includes a hand orthosis mounted to the rotating part. The hand is strapped into this supporting structure with Velcro straps. Unless the patient unstraps himself, there is no longer any chance of this hazard happening.

Probability – *Improbable/Impossible*.
Severity – *Major*.

Other not motor-related injuries

A patient might bang his hand or other part of his body into the hand rig and perhaps get a bruise. This is not likely to happen during the trials, since the patient is meant to just lie there motionless, but might happen while climbing on/off the fMRI-bed. The hand rig itself is made from hard plastic with slightly rounded edges. The only part that the patient might realistically scrape against and get hurt is a small metal part screwed on to the top of the rig, which will be just under the ceiling of the fMRI tunnel. I would speculate a patient may get a minor bruise or scratch *Occasionally*.

Probability – *Occasionally*.
Severity – *Minor*.

Now, using table 5-3, from chapter 5 (risk level lookup table) we can cross the hazard severity against its probability to get the final risk level of the different hazards.

	Severity	***Probability***	***Risk level***
Bent too far	Major	Improbable/Impossible	1
Bent too harshly	Major	Frequent	2
Jerking back & forth	Major	Improbable/Impossible	1
Stuck between moving parts	Major	Improbable/Impossible	1
Bumping into hand rig	Minor	Occasional	1

Table 6-1. *Hazards and their probability assessments.*

All hazards except one are level 1 risks. The one hazard that is not, can be avoided by using the feedback from the optical sensor. The force limit safeguard (chapter 5.6.1) can be used to halt moves that encounter too much resistance. The trick is to determine how much resistance is too much. I would suspect that using a very low threshold would make it virtually impossible to injure a patient. Another alternative would be simply to exclude such patients from the group of subjects. This would make this too a level 1 risk.

While observing these precautions, the system should be safe for use.

6.5 Patient Trials

On Jun 8, 2004 the first real patient trial was made, in the simulated fMRI environment in the lab. The mechanical operation of the system was flawless. The setting up of the entire system takes some time, especially since we had to attach EMG electrodes to measure the patient's muscle activity. The optical sensor was connected to a separate computer running Zoom, the data capture software that was used in earlier projects.

On Jun 10, 2004 we tried the system out in the fMRI room for the first time, with Anders Fagergren acting as patient. Setting up there was a great deal harder to do than in the lab. The motor bench must be placed up against the wall where the holes are, but in order to do this you need to crawl under the fMRI control table, which is more or less fixed in place. Also, the tension in the wires becomes very difficult to control with the present arrangement, as only one of them can be easily adjusted. This should be fixed before real research tests are made.

7 Conclusions

In this chapter I briefly describe the success of the project and how the system can be improved in the future.

7.1 Accomplished Goals

Almost all the goals were eventually reached, with the most notable exception being the lower maximum rotation speed. As the motor runs unreliably at the highest intended speed, we had to make do with the somewhat lower 236 degrees/second (original target was over 300). After discussing this with the people responsible for the planned studies, they felt that 236 deg/sec was fast enough. After all, it is a very quick motion, and these patients will have reflex actions at lower speeds than that.

7.2 Future Extensibility & Unresolved Issues

The computer with its RTOS can readily be used for other real-time applications. It is equipped with two DAQs that can be used both for sampling and sending data, digital as well as analog, with a resolution of 16 bits. The creation of further applications is just a question of programming, either by writing entirely new code or by modifying existing programs.

The delay in the wires was not compensated for. We do not have any method for measuring the tension in the wires, of which the delay is dependent. For future projects, if the delay is to be accurately measured, a way of deciding the tension is needed.

Also, to compensate for the delay requires a dynamically changeable motor cycle period, since a smooth function will likely be necessary, not the discrete steps that are available in the current system. At one point during the development we tried to switch to dynamically determined cycle periods. This would invariably cause the computer to lock up instantly. Being a bit short on time and wanting to leave behind us a working system, we abandoned that idea in favor of fixed cycle periods (it later turned out that the computer had some defective capacitors that would cause the system to crash).

At the time of writing, both wires are tightened with one knot. To make the tightening easier and more adaptable there should be a knot on each wire. This is easy to do and should be fixed before the system enters full use. The current configuration is way too awkward when the hand rig is placed inside the fMRI unit, as it cannot be reached easily.

Some parts of the design are a bit weak, like the motor axle, made from a piece of plastic tubing, and the plastic ball bearings found in all the caster wheels. If strong wire tension is needed in future projects, these should be replaced by non-magnetic metal ones (stainless steel).

The mechanical part of the system involves many components, most from the previous project, used without any modifications. Some are made of wood and will just barely fit together, so it takes a bit of tinkering to set up the system. Also the electrical components has added their fair share of electrical wiring.

As a last minute unforeseen modification to the system, the real-time computer was required to send position data from the digital encoder to the computer running Zoom. Otherwise it will be difficult to synchronize the measured data with the actual motion. The sending of the positional data is implemented, but the cable connection remains to be built. This should be easy; figure out which input channel on the Zoom computer to use, and solder the cable to a suitable male/female connector.

7.3 Final Words

The project was an overall success and we had a lot of fun building it. Karolinska is satisfied with the system and it hit most of its goals to an adequate level. The system is ready for use on real patients, but there are some minor modifications that could be done to facilitate its use, such as tying knots on both wire. Also, for safety reasons perhaps something should be done about the bundle of cables running from the computer to the motor/encoder. They are easy to trip over.

However, the optical sensor will likely be connected to a separate data logging computer instead of the real-time computer. There are good and bad aspects of this. It is good that the researchers use software that they know well and that is obviously more refined than what we created. The negative part is that since the real-time computer no longer receives any feedback from the sensor, the force limit safeguard will not set in.

References

[1] Skoglund, L., Björklund, M. 2003. Precision Grip Servo, *Department of Machine Design, KTH, 2003.*

[2] Dedicated Systems Experts, What makes a good RTOS, *RTOS Evaluation Project, Jun 2001.*

http://www.omimo.be/encyc/buyersguide/rtos/what_good_rtos_abstract.htm

(free registration required)

[3] Curley, Charles. 1999, Open Source Software for Real-Time Solutions, *Linux Journal, Issue 66, 1999.*

[4] Wright, C. W., Walsh, E. J. 1999. Hunting Hurricanes, *Linux Journal, Issue 58, 1999.*

[5] Timmerman, Martin, 1999, RTOS Market Survey Preliminary Results, *Dedicated Systems Magazine 1999, quarter 1, pp 6-8.*

[6] Timmerman, Martin, 1999, RTOS Market Overview – A follow up, *Dedicated Systems Magazine 1999, quarter 2, pp 6-8.*

[7] Bernat, G., Burns, A., Llamosí, A.1999. Weakly Hard Real-Time Systems, *IEEE Transactions on Computers, Sep, 1999.*

[8] Hilton, E. F., Yodaiken, V. 2001. Real-Time Applications with RTLinux, *Embedded Linux Journal, Jan 2001.*

[9] The Linux Control and Measurement Interface, http://www.comedi.org

[10] Natanaelsson, J. 2004. Design and Implementation of a Computer Controlled System for Medical Experiments on the Human Wrist, *Department of Machine Design, KTH, 2004.*

[11] Bartoo G. 2003. Risk management. *IEEE Engineering in Medicine and Biology Magazine, 2003 Jul-Aug;22(4):166-70, 172.*

Appendix A

Short overview of a few RTOS on the market

Most of the information in this overview comes from evaluations made by Dedicated Systems Experts, in what they call their 'RTOS Evaluation Project'. If you should want a more detailed view of a specific RTOS you should visit DSEs web page, http://www.omimo.be/Encyc/*, where evaluations can be downloaded for free and studied in detail (requires a free registration). They are around 80-90 pages apiece.*

eCOS
RedHat
Primarily for embedded applications, small in size.

INTIME
RadiSys Corporation Inc.
9,900 USD, plus run-time license 250 USD
+ Stand-alone operativsystem
+ Protection between NT and Intime-threads
+ Technical support
- Code Wizard generates chaotic code
- System clock manager causes delays
- Semaphore-queue management could be more effective

LynxOS
LynxWorks
3,950 USD.
Full scalability and preemption, strict adherence to standards, POSIX, multi-threaded.

OSE
Enea Data Systems
Unusable since it doesn't support the current processor (actually none in the x86-family). Primarily for embedded systems.

pSOS
Integrated Systems Inc.
14-20,000 USD, plus run-time license of a few USD.
+ Predictable
+ Completely integrated development environment
- Serious bugs
- Unclear documentation makes development hard
- No mutex

QNX
QNX Software Systems Inc.
5,000 USD, but a free version exists for non-commercial projects.
+ Good performance, fast
+ Excellent architecture for distributed and robust systems
+ Good support for different platforms
- Slow Integrated Development Environment (IDE)

RTLinux
FSMLabs
RTLinux/Pro is not free, but there is a free version, RTLinux/Free. Since standard Linux is used as the development environment, help is available on the net.

RTX
Venturcom Inc.
4,900 USD
+ Ok performance
+ Development environment nicely integrated with Windows NT
- Unpredictable object- and memory handling – only for static applications.
- Debugging requires kernel-level knowledge.

Tornado/VxWorks
Wind River
The most widely used RTOS for embedded systems. The cheapest version costs about 8,000 USD.
+ Good memory protection
- Lousy documentation
- Long worst-case response time for external interrupts (on x86)
- Serious bugs in the interrupt handling and TCP/IP network performance

µc/OS
Micrium
Free for 'peaceful' research, primarily for embedded systems, scalable, the kernel can be scaled down to as little as 2kB

Windows CE .NET
Microsoft Corporation Inc.
1,000 USD
+ Extensive platform support
+ Generally good real-time performance
- Documentation is inadequate for such a complex system